***Unidad 3 - Conociendo las pruebas***

***Objetivos de las pruebas***

***Introducción a la unidad 3.***

En esta etapa, aprenderemos sobre la importancia de las pruebas de software, su clasificación y los criterios de una buena prueba. Estos temas son fundamentales para comprender el proceso de testeo de software y su impacto en la calidad de los productos.

Estudiar las pruebas de software es fundamental por diversas razones, como garantizar la calidad del software, satisfacer al usuario, ahorrar costos, mantener la reputación de la empresa, cumplir con requisitos, optimizar recursos, trabajar en entornos ágiles, garantizar seguridad y fiabilidad, fomentar la mejora continua y destacarse profesionalmente.

* Garantizar la calidad del software: Las pruebas son esenciales para detectar defectos y problemas en el software antes de que llegue a mano de los usuarios, ayuda a garantizar que el software funcione correctamente y cumpla con sus objetivos
* Satisfacción del Usuario: Un software de calidad proporciona una experiencia positiva al usuario. Produce satisfacción a las necesidades del usuario.
* Ahorro de Costos: Detectar y corregir defectos en las etapas iniciales del desarrollo es mucho más económico que hacerlo después del lanzamiento. las pruebas efectivas ayudan a reducir costos a largo plazo.
* Reputación de la empresa: Un software defectuoso puede dañar la reputación de la empresa. las pruebas adecuadas ayudan a mantener una buena reputación y confianza de los clientes.
* Cumplimiento de requisitos: Estudiar los tipos de pruebas permite asegurarse de que el software cumpla con los requisitos especificados. Esto es esencial en sectores como la industria médica o financiera, donde el cumplimiento de normativas es obligatorio.
* Optimización de recursos: Aprender sobre las pruebas ayuda a optimizar el uso de recursos humanos y tecnológicos. Permite centrarse en áreas críticas del software y priorizar las pruebas más relevantes.
* Desarrollo Ágil: En entornos de desarrollo ágil, las pruebas son una parte integral del proceso. Comprender los principios y tipos de pruebas es esencial para trabajar de manera efectiva en equipos ágiles.
* Seguridad y fiabilidad: En el caso de aplicaciones críticas para la seguridad o la vida de las personas, como sistemas médicos o de transporte, las pruebas son esenciales para garantizar la seguridad y la fiabilidad del software.
* Mejora Continua: Estudiar pruebas de software implica aprender a identificar áreas de mejora en los procesos y en el propio software. La mejora continua es un principio fundamental en la industria del desarrollo de software.
* Competitividad profesional: Para los testers de software, tener un conocimiento sólido en pruebas es esencial para destacar en la industria y avanzar en sus carreras.

El conocimiento en pruebas de software es crucial para la calidad y eficacia en un mundo tecnológico. Ayuda a los profesionales a sobresalir en la industria y contribuir al éxito de las organizaciones con productos de calidad.

***La esencia de las pruebas de software: Un análisis integral***

Pruebas de Software: Una Exploración Inicial

Las pruebas de software son fundamentales para evaluar la funcionalidad de un programa mediante casos de prueba específicos. Buscan descubrir posibles fallas en la implementación, calidad y usabilidad del software, con el objetivo de garantizar su correcto funcionamiento y satisfacer las expectativas del usuario. Estas pruebas validan que el software cumple con los requisitos, detectan y corrigen errores, aseguran su confiabilidad, seguridad y eficiencia, y verifican su comportamiento esperado.

Es crucial que los ingenieros de software comprendan los principios que guían las pruebas de software, ya que estas son esenciales para asegurar la calidad del software, detectar y corregir errores, mejorar la usabilidad y la experiencia del usuario, optimizar el rendimiento y la eficiencia, garantizar la seguridad, cumplir con normativas y estándares, alinearse con metodologías ágiles, desarrollar una carrera especializada, reducir costos y tiempos de desarrollo, y generar confianza en el cliente y una buena reputación empresarial.

* Asegurar la calidad del Software: El testing de software es esencial para garantizar que el software funcione correctamente y cumpla con los requisitos establecidos. Esto asegura la satisfacción del cliente y evita problemas costosos en la producción.
* Detectar y corregir errores: Los testers son responsables de identificar errores y defectos en el software antes de que llegue a los usuarios finales. La detección temprana y la corrección de estos problemas son esenciales para un producto de calidad.
* Mejorar la Usabilidad y la Experiencia del usuario: Las pruebas de usabilidad y pruebas de aceptación garantizan que el software sea fácil de usar y cumpla con las expectativas de los usuarios. Una experiencia de usuario positiva es fundamental para el éxito del software.
* Optimizar el Rendimiento y la Eficiencia: Las pruebas de rendimiento y carga ayudan a identificar y solucionar problemas de velocidad y eficiencia en el software. Esto es crítico en aplicaciones que manejan grandes volúmenes de datos o usuarios.
* Asegurar la seguridad: Las pruebas de seguridad son esenciales para identificar vulnerabilidades y posibles amenazas en el software. Esto es especialmente importante en aplicaciones que manejan información sensible.
* Cumplimiento de normativas y estandares: Muchas industrias tienen regulaciones y estándares específicos que los software deben cumplir. Los testers garantizan que el software esté en conformidad con estas normativas.
* Alineación en Metodologías Ágiles: Con la creciente adopción de metodologías ágiles como Scrum, las pruebas se integran en todo el ciclo de desarrollo. Los testers son fundamentales para garantizar que el software se ajuste a los principios ágiles.
* Desarrollo de carrera: El testing de software es una carrera especializada en crecimiento constante. Los profesionales con habilidades en pruebas de software son altamente demandados y tienen oportunidades de carrera sólidas.
* Reducción de costos y Tiempos de desarrollo: Detectar y corregir errores en etapas tempranas del desarrollo es más rentable que hacerlo en la fase de producción. Esto ahorra tiempo y recursos.
* Confianza del cliente y reputación de la empresa : La entrega de software de alta calidad a tiempo aumenta la confianza del cliente y mejora la reputación de la empresa en el mercado.

En resumen, las pruebas de software desempeñan un papel crítico en la búsqueda de la calidad y el correcto funcionamiento del software, asegurando que cumpla con los requisitos, sea confiable, seguro, eficiente, y brinde una experiencia de usuario positiva. Además, son esenciales para detectar errores, mejorar la usabilidad, optimizar el rendimiento, garantizar la seguridad, cumplir con normativas, alinearse con metodologías ágiles, desarrollar una carrera especializada, reducir costos y tiempos de desarrollo, y generar confianza y reputación en la empresa.

***Principios de las pruebas de Software:***

Los principios fundamentales de las pruebas de software incluyen: las pruebas exhaustivas no son posibles debido a las infinitas combinaciones de entradas, detectar defectos tempranamente es más rentable, y las pruebas no garantizan la ausencia total de errores. Estos principios guían la creación de casos de prueba efectivos y contribuyen a mejorar la calidad del producto durante todo el proceso de desarrollo.

***Etapas involucradas en las pruebas de software:***

Las pruebas de software pasan por etapas clave: planificación, ejecución e informe. Tipos incluyen pruebas funcionales (unitarias, integración, aceptación) y no funcionales (rendimiento, seguridad, usabilidad). Son fundamentales para detectar defectos y garantizar calidad. Testers juegan papel esencial en mejorar confiabilidad y rendimiento. Conocer principios y etapas en pruebas es crucial para realizarlas con eficacia.

***Objetivos de las pruebas del Software VIDEO 01***

* Para evitar defectos mediante la evaluación de productos de trabajo tales como requerimientos, historias de usuarios, diseño y código.
* Para verificar si se han cumplido todos los requerimientos especificados.
* Para verificar si el objeto bajo prueba está completo y validar si funciona como los usuarios y otros lo esperan.
* Crear confianza en el nivel de calidad del objeto bajo prueba.
* Para encontrar defectos y fallas, y así reducir el riesgo de calidad inadecuada del software
* Proporcionar información suficiente a los usuarios para permitirles tomar decisiones, especialmente con respecto al nivel de calidad del objeto bajo prueba.
* Para cumplir con los requisitos o estándares contractuales, legales o reglamentarios y verificar el probar el cumplimiento del objeto con tales requisitos.
* Los objetivos de las pruebas pueden variar, dependiendo del contexto del componente o sistema que se está probando, el nivel de prueba el modelo de ciclo de vida de desarrollo de software.

Estas diferencias pueden incluir, por ejemplo:

* Durante la prueba de componentes, un objetivo puede ser encontrar la mayor cantidad de fallas posibles para que los defectos subyacentes se identifiquen y corrijan temprano.
* Otro objetivo puede ser aumentar la cobertura de código de las pruebas de componentes.
* Durante las pruebas de aceptación un objetivo puede ser confirmar que el sistema funciona como se esperaba y satisface los requisitos.
* Otro objetivo de esta prueba puede ser proporcionar información a los usuarios sobre el riesgo de liberar el sistema en un momento dado.

***Clasificación de las pruebas: VIDEO 02***

Hay 3 categorías principales, dentro de estas categorías contienen diferentes pruebas:

* Métodos: Utilizan para probar
* Formas: En que se realizan las pruebas
* Niveles: Diferentes niveles de pruebas

***FORMAS:***

* Manual : Proceso de probar un sistema de manera directa, sin la intervención de un ente que realice la ejecución autónoma de la misma.
* Automatizadas: Proceso de ejecución autónoma de las pruebas.

Diferencias: Las manuales las realizamos nosotros mismos interactuando con la plataforma que analizamos. Mientras que las automatizadas son pruebas que fueron programadas en un sistema externo y pueden ser ejecutadas sin nuestra intervención.

***NIVELES:***

* ***Funcionales:*** Se encargan de revisar todas las funciones del sistema, aquellos componentes que el usuario tiene interacción directa con las funciones.
  + Algunas pruebas: Unitarias, integración, Sanidad, Sistema, Humo (smoke), Regresión, Aceptación, End to End.
* ***No funcionales:*** se enfocan en elementos de experiencia de usuario, diseño, seguridad y calidad del sistema. Son pruebas que se realizan para dar terminación y detalle a la plataforma.
  + ***Algunas pruebas:*** Validación de Tiempo de carga, correcta utilización de las fuentes, seguridad de la plataforma, Rendimiento, usabilidad, portabilidad, estres, carga, entre otros.

***MÉTODOS:***

* ***Caja blanca:*** hace referencia al interior, es decir al código y la lógica del programa.
* ***Caja Negra:*** Representa la interfaz de usuario y todos los componentes que se encuentran
* ***Caja Gris:*** Cuando probamos ambas cajas.

***Profundizando los tipos de pruebas: VIDEO 03***

***Clasificación***

1. ***Pruebas funcionales:*** Se centran en probar la funcionalidad del sistema, basado en las funciones y prestaciones descritas en la especificación del requisito: Reglas de negocio, procesos, casos de uso, historias de usuario. Pueden ser aplicados en todo nivel de prueba, NORMA ISO 9126. Los casos de prueba son generados a través de la técnica de CAJA NEGRA.
2. ***Pruebas NO funcionales:*** Se centran en probar los atributos o características no funcionales que debe temer el sistema.
   1. Mediante
      1. La medición de los atributos establecidos por la NORMA ISO 9126 como : Fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.
      2. Los atributos no funcionales, pueden ser probados en todos los niveles de prueba a través de tipos de prueba.
   2. EJEMPLOS:
      1. **Pruebas de carga( Load Test):** Se relaciona con la medida del comportamiento de un sistema con una carga creciente, ej: el número de usuario concurrentes y números de transacciones para determinar la carga del sistema.
      2. **Pruebas de Rendimiento (Performance Test):** Proceso de pruebas para determinar el rendimiento del software y el uso de recursos, ej: que tan rápido se ejecuta una función determinada y los tiempo de respuesta.
      3. **Prueba de estrés (Stress Test):** Orientadas a evaluar un componente o sistema más allá de los límites especificados en los requisitos.
      4. **Prueba de estabilidad (Stability Test):** Se encargan de medir el rendimiento de un software durante una operación continua.
      5. **Robustez (Test for Robustness):** Evalúan el grado de un componente o sistema puede funcionar correctamente en presencia de entradas no válidas o condiciones de estrés.
      6. **Prueba de volumen (Volume Test):** El sistema está sometido a grandes cargas de datos e información.
      7. **Prueba de usabilidad (Usability Test):** Pruebas diseñadas para determinar en qué medida el producto del software es comprensible, fácil de aprender, fácil de operar y atractivo a los usuarios bajo condiciones específicas según la norma ISO 9126.
3. ***Pruebas Estructurales:*** Se centran en probar la estructura interna del programa

***Características:***

* Estas pruebas se realizan utilizando técnicas de caja blanca.
* Pueden realizarse en todos los niveles de prueba.
* Se basan en la arquitectura del sistema
* El área responsable de esta prueba es desarrollo.

1. ***Pruebas Asociadas al cambio: El objetivo es probar cuando el software ha sufrido algún cambio, estos cambios pueden ser después de detectar y corregir un defecto o asociados a una nueva funcionalidad.***

* ***Incluye las siguientes pruebas:*** 
  + ***Pruebas de confirmación (re-test): entran en función una vez que se corrigió un defecto y este debe probarse nuevamente.***
  + ***Pruebas de regresión: Comprueban que los cambios que se introdujeron al programa debido a correcciones o modificaciones no hayan creado nuevos defectos, estas prueba de regresión se debe realizar al final de todos los niveles de prueba.***

***UNIDAD 4 - COMPRENDIENDO LOS NIVELES DE PRUEBAS.***

Las pruebas unitarias son fundamentales en el curso de Tester de Software. En la cuarta unidad, se exploran los diferentes niveles de pruebas para evaluar la calidad del software. Es importante comprender la estructura y jerarquía de las pruebas, aplicar los conceptos en situaciones reales, utilizar herramientas adecuadas y colaborar efectivamente en equipo.

Estudiar los niveles de pruebas permite planificar y ejecutar pruebas de manera efectiva, garantizando la detección temprana de problemas. Aplicar los conceptos en situaciones reales mejora la calidad del software y corrige defectos antes de llegar a los usuarios finales. Familiarizarse con herramientas y técnicas específicas mejora la eficiencia y la evaluación del software.

La colaboración y comunicación efectiva son clave en el proceso de pruebas de software. Trabajar en equipo, compartir información clara y colaborar con otros mejora la eficacia de las pruebas y la calidad del producto final. Estos conocimientos y habilidades son indispensables en el campo del desarrollo de software en constante evolución y altamente demandados en la industria.

**Pruebas Unitarias: ¿Qué es y porqué hacerlo ?**  **VIDEO 01**

Es una manera relativamente fácil para mejorar la calidad del código producido. ¿Como? Ejecutar una pequeña parte del código para probar si este hace lo que ha sido escrito.

¿Por qué es importante? En empresas el 50% del tiempo de desarrollo es destinado al testeo. Resulta mejor el código y con menos errores, reduciendo el tiempo que estás depurando. El testeo unitario cuesta tiempo, pero si lo pospones hasta el final cuesta mucho más tiempo.

**Pruebas de Integración VIDEO 02**

El orden de la integración de los componentes (Funciones, clases, métodos) se basa principalmente en la arquitectura. Se van obteniendo componentes individuales probados de cada nivel de abstracción, se van integrando. Desarrollo de software iterativo, la integración de software es continua conforme se van generando los componentes.

¿Cómo se integra el software?

1. Seleccionar los componentes que van a ser integrados
2. Establecer una estrategia de integración
3. Preparar el ambiente de prueba (servidor y herramientas )
4. Asegurarse que estén todos los componentes en el repositorio en la última versión aprobada.

Hay 2 pruebas elementales de pruebas de integración

* Big Bang: Primero se prueba cada módulo de manera independiente. Y el bigbang prueba todo el programa de manera completa.
* Incrementales
  + Ascendentes: Es un programa que es usado para probar un módulo independientemente, se comienza de abajo de los módulos. Luego se sigue probando los módulos siguientes que van hacia arriba. la jerarquía iría de abajo hacia arriba.
  + Descendentes: Se comienza de arriba hacia abajo también con métodos o simuladores de métodos. Luego se reemplazan estos simuladores por los métodos reales y se vuelven a probar.
  + Híbridas

**Pruebas de Componentes VIDEO 03**

También conocida como prueba unitaria o de módulo, se centra en componentes que se pueden probar por separado.

Objetivos:

* Reducir el riesgo
* Verificar si los comportamientos funcionales y no funcionales del componente son los diseñados y especificados.
* Generar confianza en la calidad del componente.
* Encontrar defecto en el componente.
* Evitar que los defectos escapen a niveles de prueba más altos ( pruebas funcionales, integración, aceptación de usuario o producción).

Las pruebas de componentes a menudo se realizan de forma aislada del resto del sistema, según el modelo del ciclo de vida del desarrollo de software y el sistema, que pueden requerir objetos simulados, virtualización de servicios, arneses, stubs y controladores.

Las pruebas de componentes pueden cubrir la funcionalidad (por ejemplo, la corrección de los cálculos), las características no funcionales (por ejemplo, la búsqueda de pérdidas de memoria) y las propiedades estructurales (por ejemplo, las pruebas de decisión).

**Bases de pruebas incluyen:**

* Diseño detallado
* Código
* Modelo de datos
* Especificación de los componentes de las prueba unitarias.

**Objetos de prueba Incluyen:**

* Componentes, unidades o módulos
* Código y estructura de datos
* Clases
* Módulos de base de datos

**Defectos y fallas típicos incluyen:**

* Funcionalidad incorrecta (ej= no como se describe en las especificaciones de diseño).
* Problemas de flujo de datos
* Código y lógica incorrectos

**Enfoques y responsabilidades específicos:**

* Es muy importante que la prueba de componente la realice el desarrollador ya que él escribió el código.
* Los desarrolladores pueden alternar el desarrollo de componentes con la búsqueda y reparación de defectos.
* Los desarrolladores suelen escribir y ejecutar pruebas después de haber escrito el código de un componente.
* Sin embargo, especialmente en el desarrollo ágil, la escritura de casos de prueba de componentes automatizados puede preceder a la escritura del código de la aplicación.

EJ= TDD, Desarrollo basado en pruebas. El desarrollo impulsado por pruebas es altamente iterativo y se basa en ciclos de desarrollo de casos de prueba automatizados, luego construyendo e integrando pequeños fragmentos de código, luego ejecutando las pruebas de componentes, corrigiendo cualquier problema y refactorizando el código.

Este proceso continúa hasta que el componente se ha construido por completo y se pasan todas las pruebas del componente.

El desarrollo impulsado por pruebas es un ejemplo de un enfoque de prueba primero.

**Pruebas de sistemas VIDEO 04**

Comportamiento de todo el sistema o el producto que se define en el alcance del proyecto. Hasta este punto se probaron cada componente individualmente, ahora se ve al sistema como un todo.

Las pruebas se refieren a probar las pruebas funcionales (adecuación, exactitud, cumplimiento de la funcionalidad, seguridad) requisitos No funcionales (fiabilidad, usabilidad, eficiencia, mantenibilidad, y portabilidad).

Caso de prueba pueden ser obtenidos desde la especificación de los requisitos, casos de uso, proceso de negocio, evaluación de riesgo, y todo documento del sistema que tengamos.

Probamos el sistema desde el punto de vista del usuario, que se haya implementado completa y correctamente los requisitos funcionales y no funcionales, es importante que el entorno donde probamos sea similar al entorno real.

MÉTODOS a utilizar: CAJA NEGRA.

**Pruebas de aceptación VIDEO 05**

Hasta este punto se realizaron las pruebas de sistema, integración y unitarias, ya corrigió sobre defectos, y el sistema se entrega al usuario para que pueda verificar y acepte el software. Se realizarán preguntas claves

¿Se puede lanzar el sistema?

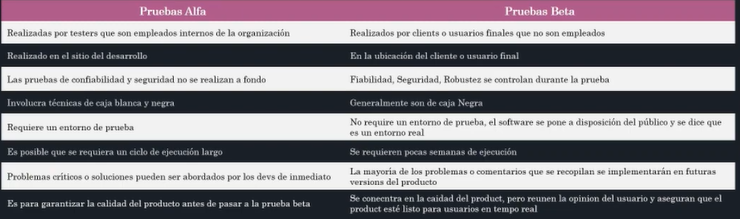
¿Cuales son ,si los hay, los riesgos (comerciales) pendientes?

¿El desarrollo ha cumplido con sus obligaciones?

La responsabilidad de las pruebas de aceptación es del cliente aunque también pueden participar otras partes de interés. EL entorno para que lo pruebe es un “Espejo” de la producción lo que se logre acá, que si es que falla algo se puede hacer un “HOTFIXED” un arreglo rápido antes que llegue a producción.

El objetivo de esta prueba es: que genere confianza en el sistema a nivel de negocio, producto y características útiles para personas, usabilidad, diseño, funcionalidad, vistoso hacia el usuario.

Primero están las pruebas Alfa y luego las betas.



Profundizacion en lso niveles de pruebas